JAIST Repository
https://dspace.jaist.ac.jp/

Title	大規模システムの分析モデルにおけるモデル間整合に関する研究
Author(s)	藤本，幸伸
Citation	
Issue Date	2002-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/1578
Rights	
Description	Supervisor:片山 卓也，情報科学研究科，修士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

修 士 論 文

大規模システムの分析モデルにおける
モデル間整合に関する研究

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

藤本 幸伸

2002年3月

修 士 論 文

大規模システムの分析モデルにおける モデル間整合に関する研究

指導教官　片山卓也 教授

審査委員主査　片山卓也 教授
審査委員　落水浩一朗 教授
審査委員　権藤克彦 助教授

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

010102 藤本 幸伸

提出年月: 2002 年 2 月

Copyright © 2002 by Fujimoto Yukinobu

概 要

大規模システムの分析モデルにおけるモデル間の整合性について、実例を用いて検証し、システムの上流過程で起きている分析モデル間の不整合について検討した。見つけ出した不整合について、計算機による支援が可能な整合性検証法を検討し、不整合を検知するための支援ツールを作成した。整合性の検証が困難な抽象度が異なるモデル間の対応関係を明確にし、整合性検証の方法を提案した。

目 次

第 1 章　はじめに **1**
1.1 研究の背景 1
1.2 目的 1
1.3 本論文の構成 1

第 2 章　ITS 車載システムアーキテクチャ **3**
2.1 概要 3
2.2 全体構成 3
2.2.1 ITS サービス 3
2.2.2 論理アーキテクチャ 5
2.2.3 物理アーキテクチャ 7
2.2.4 方式評価シート 12
2.2.5 方式採用リスト 13
2.2.6 その他 14
2.3 ITS 車載システムアーキテクチャの作成過程 14

第 3 章　モデル間における不整合の分析 **17**
3.1 各モデル間の関係 17
3.2 同じ抽象度で記述されたモデル間における不整合 18
3.3 不整合の例 20
3.4 異なる抽象度で記述されたモデル間における不整合 25

第 4 章　不整合検証法 **26**
4.1 方式モデル-個別 MFD 間の不整合 26
4.1.1 モデル間の対応関係 26
4.1.2 不整合検出の手法 27

第 5 章　抽象度が異なるモデル間の整合を検証するための制約記述法 **28**
5.1 OCL の概要 28
5.2 制御モデルと方式モデルの対応関係 29
5.2.1 OCL による制約記述 31

第 6 章　計算機支援環境 **33**
6.1　ツールの概要 33
6.2　Rational Rose から Excel へのデータ変換 34
6.3　データ比較による不整合検証 35
6.4　結果と評価 36

第 7 章　おわりに **38**
7.1　考察 38
7.2　まとめ 39

図 目 次

2.1 ITS サービス 4
2.2 情報モデル例（周辺モデル） 5
2.3 制御モデル例（[2] 経路案内） 6
2.4 物理アーキテクチャの記述例 8
2.5 MFD の記法 9
2.6 方式モデル例 10
2.7 個別モデルの例 11
2.8 方式評価シートの例 12
2.9 方式採用リストの例 13
2.10 作成手順の全体像 16

3.1 方式モデルが個別モデル上で変更された例 19
3.2 不整合の例１ 21
3.3 不整合の例２ 22
3.4 不整合の例３ 23

5.1 制御モデルの一部と方式モデルの対応 30
5.2 制御モデルの一部と方式モデルの対応２ 32

6.1 ツールの概要 34
6.2 変換前のクラス図 35
6.3 Excel データの例 36
6.4 ツールの実行例 37

第1章　はじめに

1.1　研究の背景

近年、オブジェクト指向という考え方は、ソフトウエア開発における主流の考え方になってきている。オブジェクト指向分析・設計で広く用いられるモデリング言語としてUMLがある。UMLはBooch、Rumbaugh、およびJacobsonが提唱したオブジェクト指向分析・設計方法論のモデリング言語を統一したものである 。モデリング言語は開発者がシステムを構築する際に、問題点や解決策を議論する上で重要な役割を果たす。そのひとつとして、開発者間で議論をするときの共通言語としての役割がある。ソフトウェア分析・設計を開発者共通のモデリング言語に基づいて行うことは、開発者間の誤解を減らし正しい分析モデルを作成することにつながる。しかし、現在手に入れることが可能なUMLエディタではモデルの記述や意味が形式化されておらずそのため複数の設計者によって設計されたモデル間における整合性を正確に取ることが困難である。大規模なシステムの設計においてもオブジェクト指向開発は有効な方法である。しかし大規模システムには様々な視点から作成された莫大な量の分析モデルが存在するためモデル間の整合性を保つことが大変困難である。迅速かつ正確に大規模なシステムを設計するためには分析モデル間の整合を十分に考慮することが不可欠である。

1.2　目的

本研究の目的は大規模システムの分析にオブジェクト指向開発法を適用した例としてITS[1]車載システムアーキテクチャを取り上げ、実際に大規模システムの分析モデル間の整合性を調べ、現在の開発現場において問題となっている分析モデル間の不整合を明らかにすること、不整合検証のための計算機支援環境を提案することである。また、不整合を起こしにくいモデリング法についても検討する。

1.3　本論文の構成

本論文ではまずはじめに、第2章において実例として使用するITS車載システムアーキテクチについて説明する。

[1]Intelligent Transport System

次に第3章では、ITS車載システムアーキテクチャで規定されている各モデル間で実際に生じている不整合について明らかにする。

第4章では、第3章で明らかにされた不整合を検出する手法を提案する。

第5章では、抽象度が異なるモデル間における不整合検出のための制約記述法について述べる。

第6章では、抽象度が同じモデル間の不整合検出のための計算機支援ツールを紹介する。

第2章　ITS車載システムアーキテクチャ

本章では実例として用いたITS車載システムアーキテクチャ[1]の概略を説明する。本研究において不整合個所の検証に使用した論理アーキテクチャの情報モデル、物理アーキテクチャの方式モデル、個別MFDとその作成過程に関わる事柄についてのみ説明する。

2.1　概要

ITS（Intelligent Transport Systems）高度道路交通システムは、エレクトロニクス、情報処理・通信、制御技術を交通運送分野に総合的に適用し、安全・快適・効率的な道路交通システムの実現を目指すものである。ITSシステムアーキテクチャは1996年に関係省庁によって定められた「高度道路交通システム（ITS）推進に関する基本構想」に基づき、ITSで必要とされる機能を基本要素ごとに分割整理し、それらの関連を明らかにすることによって基本要素単位や分野ごとに開発を進める際に機能的な相互連携を確保し、ITSの円滑、効率的な発展を目指すために構築された。ITS車載システムアーキテクチャはITSシステムアーキテクチャのうち、民生的な見地から要求仕様が決定される車載システムについてまとめたもので、公共的な見地から要求仕様が決定されるインフラストラクチャとの間のインターフェイスを決定するために構築された。

2.2　全体構成

2.2.1　ITSサービス

ITS車載システムアーキテクチャを検討していく上では、その前提として実現すべきITSサービスを想定することが必要である。そのため、「高度道路交通システム（ITS）の全体構想」（関係5省庁、1996年7月公表）をもとに19のITSサービスと45のサブサービスが設定されている。表2.1にITSサービスおよびサブサービスの一覧を示す。

ITS車載システムアーキテクチャ	
ITSサービス	サブサービス
1．ナビゲーション/動的経路案内	[1]自車位置情報提供
	[2]経路案内
	[3]経済走行支援
2．道路交通情報提供	[4]道路交通状況情報提供
	[5]規制情報提供
	[6]標識情報提供
3．サービス情報提供	[7]サービス情報提供
	[8]サービス予約支援
4．自動料金収受	[9]有料道路自動料金収受
	[10]施設･サービス料金自動収受
5．資格支援	[11]視覚補強情報提供
	[12]視覚情報提供
6．走行環境情報提供･警報	[13]走行環境情報提供
	[14]周辺障害物情報提供
	[15]走行状況提供
	[16]違反警報
7．社内監視･警報	[17]車両監視･警報
	[18]ドライバ覚醒･監視
	[19]救急センタ通報
8．走行制御	[20]駐車場進入退場支援
	[21]前車追従走行
	[22]車線維持走行
9．自動運転	[23]専用レーン合流
	[24]専用レーン自動走行
	[25]隊列走行
10．高度交通量制御	[26]旅行時間制御
	[27]中央決定型経路誘導
11．道路維持車両管理	[28]道路維持車両運行管理
12．特殊車両運行管理	[29]特殊車両通行監視
13．マルチモーダル利用支援	[30]マルチモーダル利用計画支援
	[31]マルチモーダル利用動的案内
14．公共交通運行･運行管理支援	[32]バス運行管理支援
	[33]タクシー運行管理支援
	[34]公共車優先走行支援
	[35]公共交通料金自動収受
15．商用車量運行管理支援	[36]運行管理支援
	[37]動的配車･配送支援
	[38]荷物管理支援
16．歩行者等危険防止	[39]歩行者等危険防止
17．緊急時自動通報	[40]緊急時通報支援
	[41]盗難車追跡
18．緊急車両救援活動支援	[42]緊急車優先支援
	[43]緊急車運行支援
19．高度情報通信社会対応支援	[44]マルチメディア情報提供
	[45]グループ車両間情報交換支援

図 2.1: ITS サービス

2.2.2 論理アーキテクチャ

論理アーキテクチャは ITS に要求されるサービスを分析し、その実現に本質的に必要とされる機能と情報を、その実現に立ち入らずに示すものである。論理アーキテクチャは情報モデル、制御モデルという形でまとめられており、UML[1]を一部アレンジした形で記述されて いる。

情報モデル

情報モデルは論理アーキテクチャの記述に必要となる概念とその間の関係を示したものであり、クラスダイアグラムで記述されている。情報モデルは論理アーキテクチャに出現する登場人物とその関係の記述であり、サービスそのものを直接的に記述するものではない。図 2.2 は情報モデルの代表的な例である。

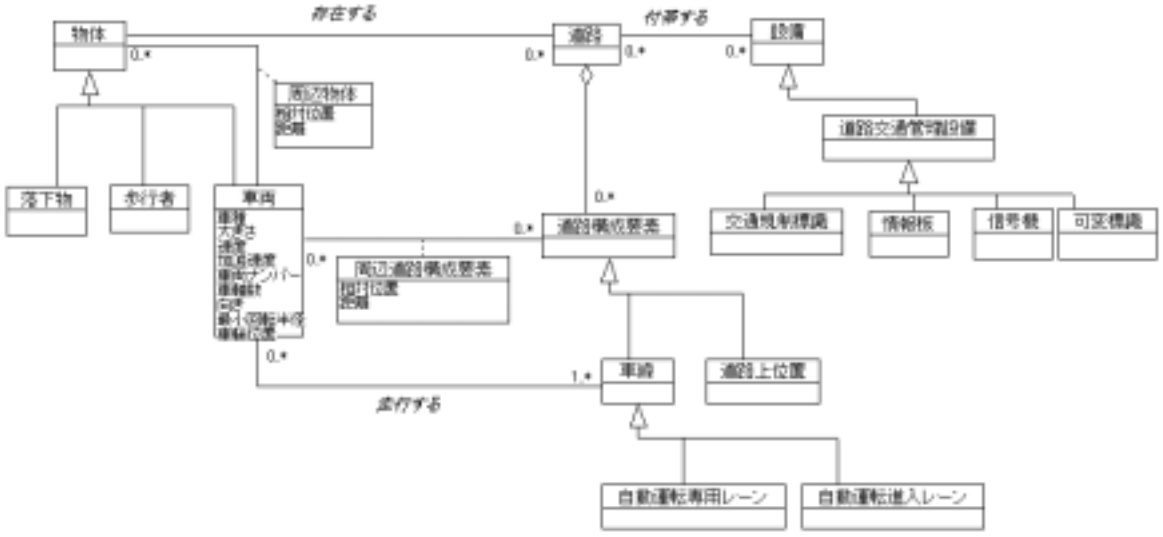


図 2.2: 情報モデル例（周辺モデル）

- 車両の周辺には様々な物体が存在する。物体には落下物や車両等が含まれる。車両と道路上に存在する物体（車両や歩行者を含む）との関係によって、車両にとって相対的に周辺物体が定義される。
- 車両は速度や加減速度等の物理的な特性を保持する。
- 同様に道路構成要素との間にも、相対的に周辺道路構成要素が定義される。

[1]Unified Modeling Language

- 道路には各種設備が付帯している。

制御モデル

制御モデルはサービスごとにクラスダイアグラムで記述され、情報モデルで記述されたクラスがどのように関わりながら該当するサービスを実現するのかを示している。制御モデルは特定のサービスの実現方法を示すものではなく、該当サービスを実現するにあたり、クラスがどのような役割を持っているかを示すものである。情報モデルは繰り返し出現する役割を５つパターンで表現している。「検知収集」、「情報提供」、「指示制御」、「通知警告」、「確認」がそのパターンである。図 2.3 は制御モデルの例である。



図 2.3: 制御モデル例（[2] 経路案内）

- 「ドライバ」は、「02C 検知収集」に対して、「目的地」と「経路条件」を「情報提供」する。
- 「02C 検知収集」は「車両」の「位置」を「検知収集する」。同時に、リアルタイム情報として、「道路」の「渋滞状況、事故状況」、「経路」の「リンク旅行時間」、および「交通規制情報」をそれぞれ「検知収集」する。また、「道路網」データも収集する。
- 「02C 経路計算」は、「02C 検知収集」が収集した情報に基づいて「推奨経路」と「到着予想時間」を計算する。

- 「02C 情報提供」は、「位置」「推奨経路」「等 y 宅予想時間」を、「02C 地図情報」から収集した周辺の「地図情報」と共に、「ドライバ」に「情報提供」する。
- 「02C 経路計算」は、定期的に「到着予想時間」を、また「道路」の「渋滞状況、事故状況」が変化したときには、「推奨経路」および「到着予想間」を再計算し更新する。

2.2.3 物理アーキテクチャ

物理アーキテクチャは ITS の実現構造を示すものである。実現構造とは論理アーキテクチャから導き出される機能や情報が車両や路側等にどのように配置され、それらの間にどのような情報や制御の流れが存在するかを示すものである。実現構造は、ITS の各サービスを実現するために本質的に必要とされる機能や情報の配置であり、特定のシステムの実現に際して必要とされるすべての機能や情報を示すものではない。配置される機能や情報は、それを実現するために必要となる具体的な実現技術の存在を想定しているが、必ずしも特定の実現技術と 1 対 1 に対応するものではない。物理アーキテクチャは方式モデル、個別 MFD、全体 MFD で構成される。

アーキテクチャの表現方法

物理アーキテクチャはサブシステム、ユニット、メッセージフローの組合わせで表現される。実現構造を示すにあたり、まず機能や情報の配置個所として、車両、路側、センター、外部がある。これらの４つを最上位サブシステムと呼ぶ。つぎに、最上位サブシステム上に配置される機能や情報の実現単位をサブシステムもしくはユニットと呼ぶ。サブシステムはユニットよりも抽象度の高い機能や情報の集まりを示している。ただし、サブシステムやユニットは必ずしも特定の実現技術と１対１に対応しているとは限らず、また具体的な装置やソフトウエア等の実装や実現を想定したものではない。ユニットやサブシステム間の情報や制御の流れを表したものがメッセージフローである。図 2.4 に物理アーキテクチャの例を示す。

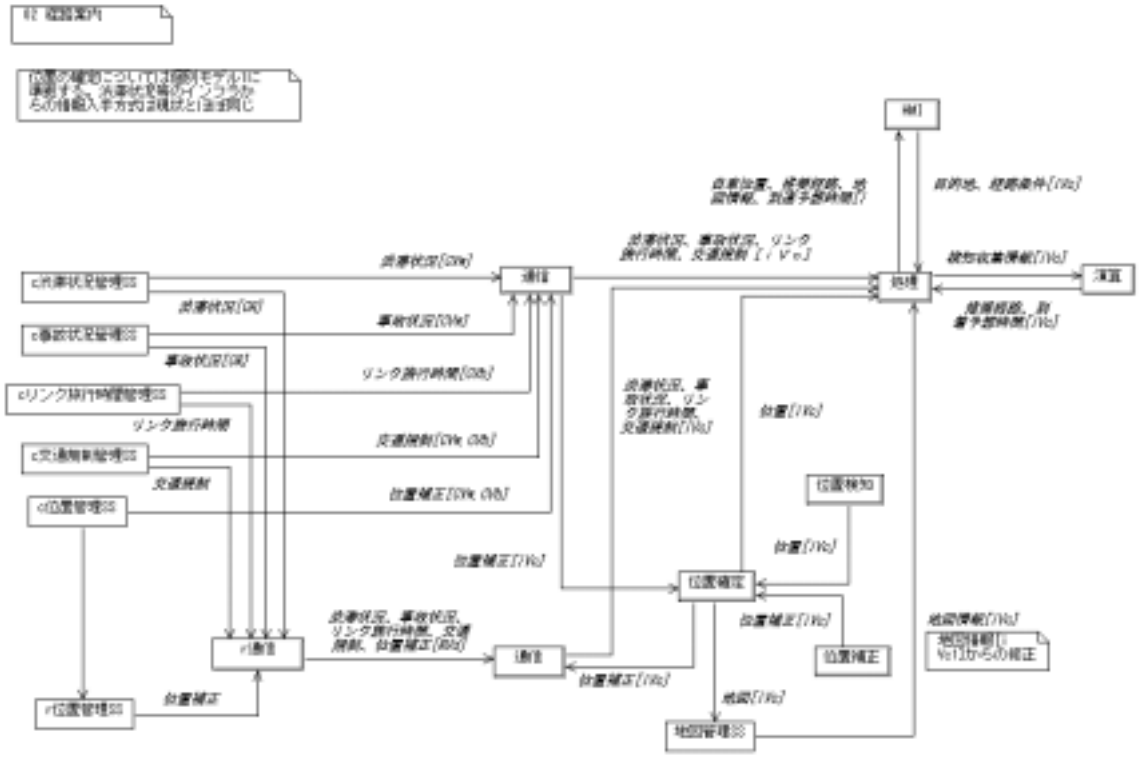


図 2.4: 物理アーキテクチャの記述例

- 最上位サブシステムはプリフィックスで表現される。プリフィックスは外部（e)、センター（c)、路側（r）であり、プリフィックスが無いものは車両を表す。

MFD の記法

方式モデル、個別 MFD、全体 MFD は、MFD[2] として、同様の記法で記述されている。これらはの MFD は図 2.5 の記法に基づいて記述されている。

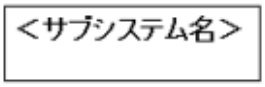
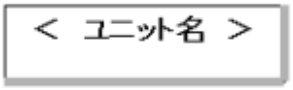
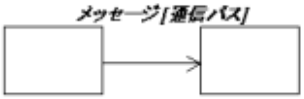
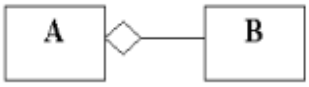
	記号	説明
サブシステム		サブシステムを示す。サブシステム名はその配置位置を示す以下のプリフィクスを持ちうる。外部(e)、センター(c)、路側(r)。プリフィクスを持たないサブシステムやユニットは車両への配置を意味する。
ユニット		ユニットを示す。サブシステム同様のプリフィクスをもちうる。
メッセージフロー		メッセージの流れを示す。通信パスを示記号(通信リファレンス中で定義された記号)を[ ]内に示すことができる。
集約関係		サブシステムもしくはユニット Bがサブシステムもしくはユニット Aに包含されることを示す。

図 2.5: MFD の記法

[2]Message Flow Diagram

方式モデル

方式モデルは論理アーキテクチャの制御モデルで抽出した重要な機能や情報についてその機能や情報管理の実現方式を MFD でモデル化したものである。方式モデルは複数の制御モデルで共通に出現するもの（共通方式モデル）と個別の制御モデルにのみ出現するもの（個別方式モデル）がある。方式モデルは制御モデル中の１つの機能に対して考えられるすべての実現方式をモデル化するため、１つの機能に対して複数の方式モデルが方式案として作成されている。図 2.6 は [1] 自車位置情報提供の制御モデルの「車両の位置情報を検知収集する」という機能の実現方式を表した方式モデルの例である。図の下側のモデルは、この方式モデルが対応している制御モデルの一部分を抜き出したものである。

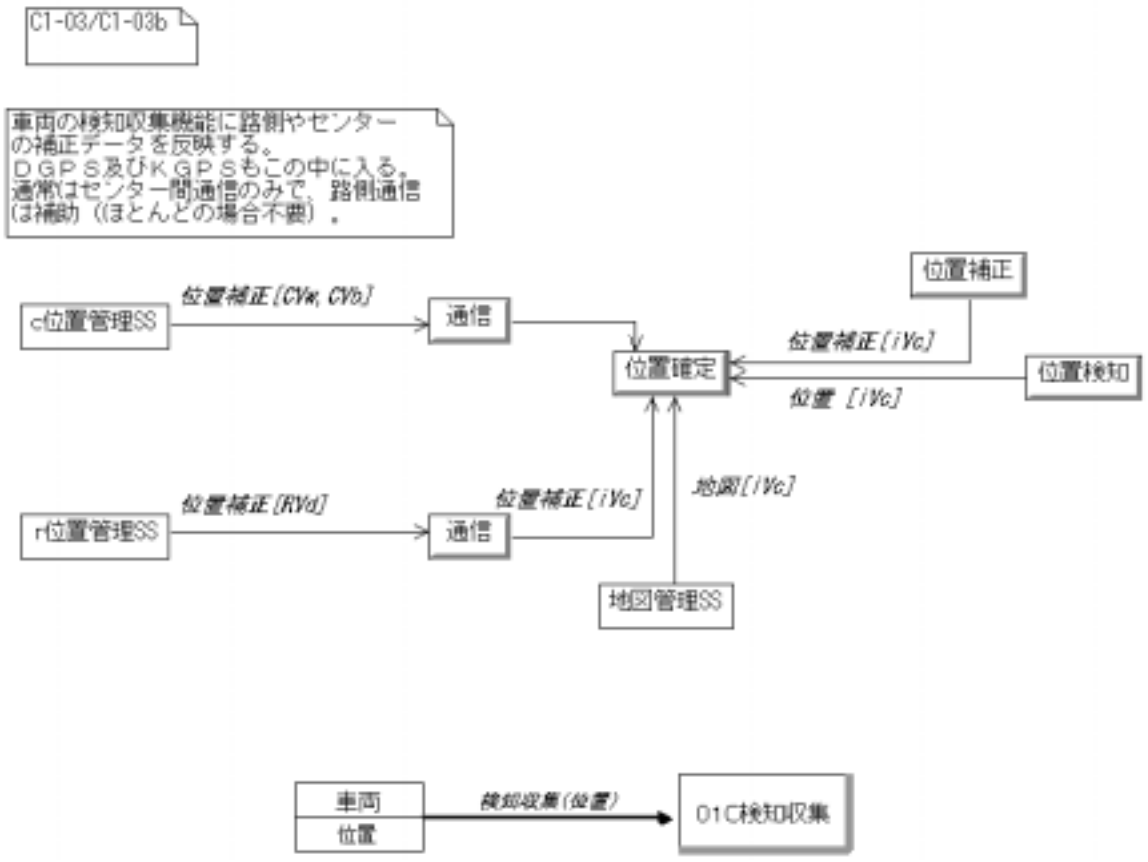


図 2.6: 方式モデル例

個別 MFD

個別 MFD はサブサービス毎に、そのサブサービスの実現方式を表したモデルである。サブサービスの実現に必要となる機能や情報管理の実現方式は方式モデルから選択する。選択の根拠は方式採用リストに示されている。採用された実現方式に基づいて、各サービスに対応する個別 MFD が作成されている。図 2.7 は個別モデルの例である。

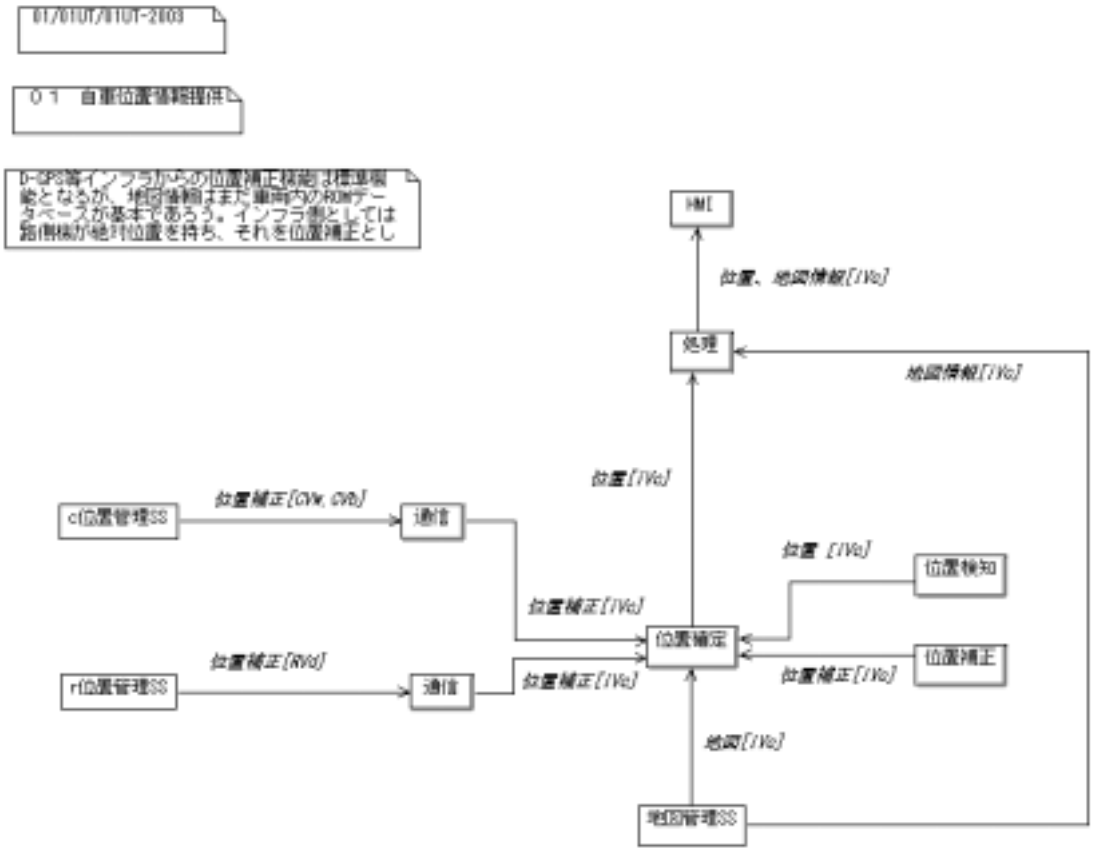


図 2.7: 個別モデルの例

個別 MFD と制御モデルは同一のサービスを表現するモデルであるという点で対応するが制御モデルのすべての構造が方式モデルでモデル化されているわけではないので、採用されたすべての方式モデルを統合したものと個別モデルは一致しない。採用されたすべての方式モデルを統合したものは個別 MFD のサブセットとなるため方式モデルの構造は必ず個別 MFD に現れている。

全体 MFD

全体 MFD は全サービスの個別 MFD をすべて重ね合わせて１つの MFD として表現したものである。全体 MFD は ITS 全体の実現に必要とされる機能や情報の配置と、その間の制御や情報のフローを示すものである。全体 MFD では中間サブシステムが作成されている。中間サブシステムは複数のユニットやサブシステムを包含したもので、システム全体への網羅性や重要性を検討して数十個作成されている。

2.2.4 方式評価シート

制御モデルで抽出した機能の中の特定の機能に対する複数の方式案を定性的に比較し、特性付けするもので、方式案毎に作成されている。図 2.8 に方式評価シートの例を示す。

C1-01 「地点」の（気象、レストラン等）「サービス情報」を「検知収集」する

評価項目 / 方式案		性能・機能						コスト		リスク	通信		計	判断理由、備考（2003年）	コメント（将来動向（2008年／2018年））
		システムの柔軟性・拡張性	安全性	データの更新性	セキュリティ	信頼性	サービスの平等性	インフラコスト	ユーザコスト	技術的実現性	システムボック選別と重要性	レスポンス	信頼性	合計点（参考）		
－	評価項目の選択と重み付け	2	0	3	0	2	2	2	3	2	0	2	0	－	信頼性のある最新のサービス情報をより安く入手することが重要と判断し、評価項目の重み付けをおこなった。	
案a	車載データベースからのサービス情報を検知収集する	×		×		○	×	◎	◎	○		○		16	車載データベースのみでは動的なサービス情報が得られないことから、更新性と柔軟性を×とした。	2003年以降、携帯電話を使ったオンデマンド型に移行していくと推測。
案b	外部機関から情報をセンターで入手し、利用可能なメディアで車両に供給する	◎		◎		◎	○	○	×	○		×		20	サービス情報を入手するには一番良い方式と判断した。ただし、ユーザコストは高くなるので×とした。	サービス費、通信費の低下により2003年以降の中心的なモデルとなる。
案c	外部機関から情報をセンターで入手し、路側を経由して車両に供給する	○		◎		◎	×	×	×	○		×		14	路側にあるビーコン等を必ず使用しなければならないことからインフラコスト、平等性を×とした。	全ての種類の情報を、好きなときに取れないと言う欠点がある。2018年でもこの状況は変わらないと推測。
案d	外部機関から利用可能なメディアで直接車両に供給する	◎		◎		×	○	○	×	○		○		18	サービス情報の性格上、外部機関から情報を車両に取り込む案も採用。情報の信頼性は×と評価した。	外部から限定された種類の情報を入手することは2003年以降も続くと判断。
案e	センターで地点のサービス情報を検知収集する	◎		◎		◎	○	○	－	－		－		18	この案単独で評価を行うことはあまりにも部分的過ぎて難しいため、他案の一部に使われる事を前提として評価した。	センターを経由し、車両に特化した情報を提供する傾向は2003年以降も続くと判断。

図 2.8: 方式評価シートの例

2.2.5　方式採用リスト

方式採用リストは制御モデルで抽出した機能の実現方式を方式案の中から選択するために作成されたリストである。通常方式モデルは方式案として１つの機能に対し複数作成され、サブサービスごとにもっともふさわしいと判断された方式案が採用される。方式採用リストには方式案の採用基準と判断結果が記されている。図 2.9 に方式採用リストの例を示す。

［1］自車位置情報提供

自車位置を検出し、それをドライバ(乗員)のニーズに対応した形で表示、出力するサービス。地図情報による位置の特定とともに、その周辺地域に関する情報も、地図等の各種表示や音声等によって提供を行う。

方式No.：	C1-03	方式内容：	[車両」の「位置(グルーバル)」を「検知収集」する。

案	採　否			理　由
	2003年	2008年	2018年	
a	△	×	×	精度向上が課題。
b	○	○	○	本案で地図データベースの更新性が改善されれば、最も良い。
c	×	×	×	インフラの無い所で機能しない事は問題。
d	×	×	×	インフラ側に要求されるサービス要件であり、本サブサービスでは対象外。
e	×	×	×	同上
f	×	×	×	同上

最終判断：	案ｂ：２００３年では、まだ案ａも残存する可能性があるがインフラの整備状態と実現性により案ｂと判断

方式No.：	C1-14	方式内容：	「地図情報」を「検知収集」する

案	採　否			理　由
	2003年	2008年	2018年	
a	○	△	×	現状の方式。データの更新性が課題。
b	×	○	○	本案で地図データベースの更新性が改善されれば、最も良い。

図 2.9: 方式採用リストの例

2.2.6 その他

ITS 車載システムアーキテクチャは上記のほかに以下の資料で構成されている。

- 通信リファレンス
  最上位サブシステム間に存在しうる通信パスや技術を示すもの。方式モデルから参照される。全体に１つ作成される。

- HMI リファレンス
  HMI[3]技術を示すもの。方式モデルから参照される。全体に１つ作成される。

- 通信パスマトリクス
  個別 MFD 中に出現するすべての情報や制御のフローを横断的に整理したもの。全体に１つ作成される。

- 想定シナリオ
  2003 年、2008 年、2018 年の各年代におけるサービスの展開、利用可能な技術を各種公表資料等に基づき想定したもの。

- サブシステム一覧
  物理アーキテクチャで使用するサブシステムの位置関係を階層的にまとめたもの。MFD と同様の記法で作成されている。

- ユニット一覧
  物理アーキテクチャで使用するユニットの位置関係を階層的にまとめたもの。MFD と同様の記法で作成されている。

2.3 ITS 車載システムアーキテクチャの作成過程

ITS 車載システムアーキテクチャは論理アーキテクチャ、物理アーキテクチャの順で作成される。論理アーキテクチャでは ITS で規定されているサービスの実現に必要となる機能や情報の洗いだしが行われ情報モデル、制御モデルという形でまとめられている。物理アーキテクチャでは論理アーキテクチャで作成されたモデルを基により細かい粒度で制御モデルで抽出された機能や情報の実現構造を MFD という形でまとめている。
物理アーキテクチャは以下の手順で作成される。図 2.10 に作成過程の全体像を示す。

- 論理アーキテクチャの制御モデルから機能と情報を抽出し、共通性や重要性から横断的に整理して機能の一覧を作成する。

[3]Human Machine interface

- 通信層や HMI 技術をリファレンスとして整理し、通信リファレンス、HMI リファレンスを作成する。
- リファレンスを参考としながら、先にあげた機能の実現方式を検討し、方式モデルを作成する。機能の実現方式は可能性の大きい複数の方式モデル案からなる。
- 各方式モデル案に相対的な特性付けをし、方式評価シートを作成する。
- 方式モデルに登場するサブシステムやユニットを横断的に整理し、階層構造を作り、サブシステム一覧、ユニット一覧を作成する。
- サブサービス毎に方式モデルと方式評価シートを参考にしながら方式採用リストをつくる。
- 採用した方式モデルに基づいて個別 MFD を作成する。
- 個別 MFD を集計して主要なサブシステムやユニットを表す全体 MFD を作成する。

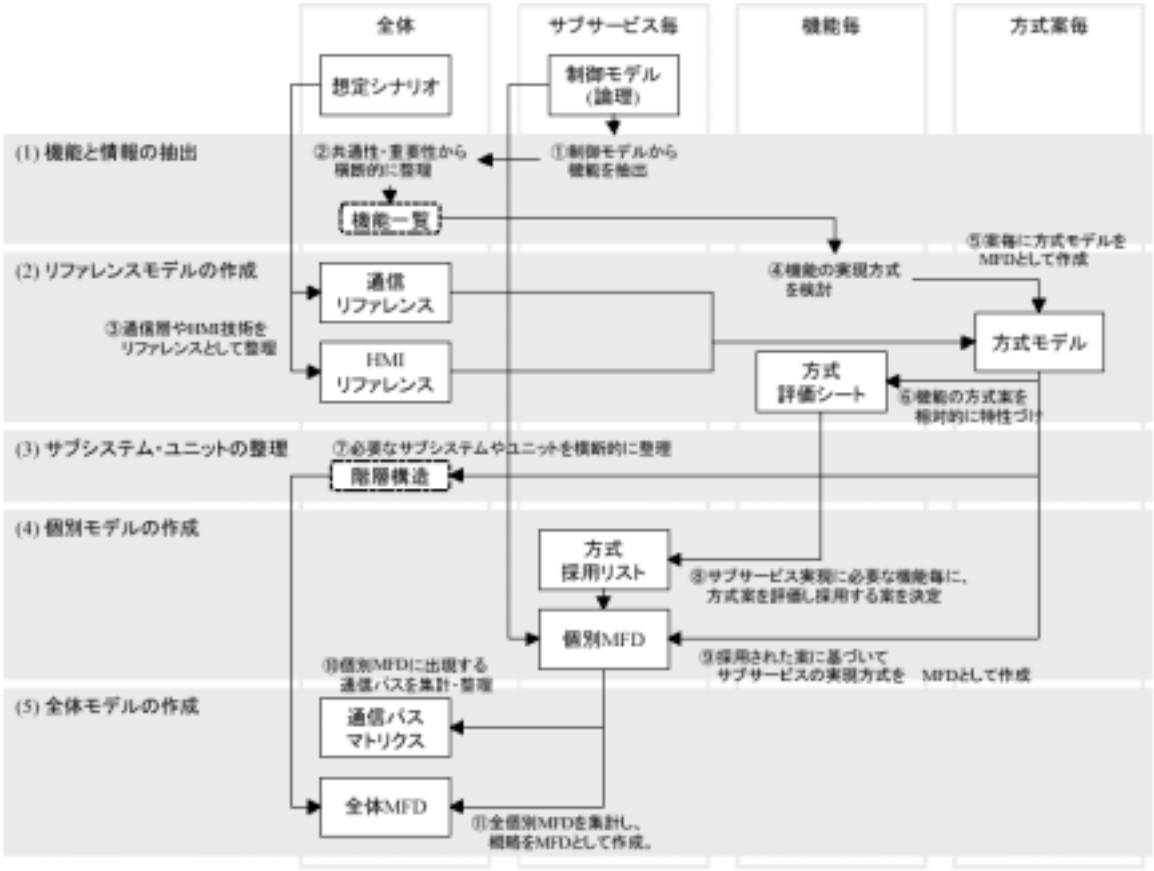


図 2.10: 作成手順の全体像

第3章　モデル間における不整合の分析

ITS 車載システムアーキテクチャは設定した ITS サービスを実現するために自動車、道路、交通システム、社会環境のような莫大な情報をコンパクトにまとめ、システムの標準化活動をはじめとした ITS の基盤を作成するための資料としての意味合いが強い。情報をまとめるための手法としてオブジェクト指向分析法を用い、分析モデルは UML を ITS に合わせてカスタマイズした記法でクラス図、MFD で作成されている。また、各サービスの実現方式をあらわすモデルは、複数の方式案から採用されたモデルを組み合わせてモデル化されている。そのため、各方式案の評価、採用を決定するための方式評価シート、方式採用リストなどモデル図以外の資料が多く存在する。本研究では、ITS 車載システムアークテクチャの分析モデルの中で、サービスの実現方式に特に関係が深い制御モデル、方式モデル、個別 MFD の間の不整合について分析した。

3.1　各モデル間の関係

整合性検証の対象とした制御モデル、方式モデル、個別 MFD についてモデル間の関係を整理する。ITS 車載システムアーキテクチャには論理アーキテクチャと物理アーキテクチャという記述の抽象度が異なる 2 つのアーキテクチャで構成されている。論理アーキテクチャはサービスに本質的に必要と思われる機能と情報をその実現に立ち入らないレベルでモデル化している。制御モデルは論理アーキテクチャのモデルである。物理アーキテクチャは論理アーキテクチャによって抽出された機能と情報を論理アーキテクチャより具体的な構成要素を用いて記述している。方式モデル、個別 MFD は論理アーキテクチャのモデルである。そのため制御モデルは方式モデル、個別 MFD より抽象度の高い視点で作成されたモデルである。制御モデルと方式モデルの関係を以下にまとめる。

- 制御モデルと方式モデルの関係
  - 方式モデルは制御モデルで抽出された機能と情報の中で、重要と思われる機能と情報管理の実現方式を表している。
  - 制御モデルと方式モデルは記述の抽象度が異なる。制御モデルはクラス図、方式モデルは MFD で書かれている。
  - 制御モデルは方式モデルより抽象度が高いモデルである。

方式モデルと個別 MFD は物理アーキテクチャに属するモデルである。物理アーキテクチャのモデルはあらかじめ規定された MFD の記法にしたがって作成されている。そのため方式モデル、個別 MFD の各モデルを構成しているユニットやサブシステムといった構成要素は同一である。個別 MFD は複数の方式モデルを統合したものを基本に作成されているが、個別 MFD で独自に追加される構成要素が存在するため方式モデルを統合したものと個別モデルは一致しない。個別 MFD は方式モデルを統合したものと比較して意味的により詳細なモデルであるといえる。また、個別 MFD の中で方式モデルに対応している部分と該当する方式モデルは一致しない場合がある。個別 MFD 上では方式モデルを、その意味が変わらない程度に変更することが許されている。図 3.1 に例を示す。図の上側は「経路案内」というサービスを表わす個別モデル、下側は個別モデルのうち、太線で書かれた部分と対応している方式モデルである。図のように、方式モデルで書かれているサブシステムの 1 部が個別モデルでは書かれていない。これは方式モデルの c 交通流情報管理 SS と c 渋滞状況管理 SS、c 道路交通情報管理 SS、交通環境情報管理 SS と c 事故状況管理 SS はそれぞれ汎化関係であるため、個別モデル上に記載しなくても意味が変わらないと判断されたためである。方式モデルと個別モデルの間のこのような記述の違いは不整合にならない。

方式モデルと個別モデルの関係を以下にまとめる。

- 方式モデルと個別 MFD の関係
  - 制御モデルと対応関係のある方式モデルの構造はすべて個別 MFD に現れる。
  - 方式モデルと個別 MFD は同じ MFD の記法で書かれた MFD である。
  - 制御モデルと対応関係のある方式モデルをすべて統合したものと個別 MFD は一致するとは限らない[1]
  - 個別 MFD の中で方式モデルと対応する部分と各方式モデルは基本的に同一の構造をしているが、設計者の判断で基本的な意味を変えない程度の変更が許されている。

3.2 同じ抽象度で記述されたモデル間における不整合

現在のモデリング法における、方式モデルと個別 MFD の間の整合条件を示す。

- 方式モデルの構造が個別 MFD に必ず現れている。
  - 方式モデルの構造はサブシステム、ユニットの各ノードとデータフローで表現されたアークの組合わせである。

[1] 方式モデルは制御モデルで抽出された機能と情報の中で重要と思われるものについてのみ作成されるため、制御モデルで抽出されたすべての機能と情報管理の実現方式を表しているとは限らない。

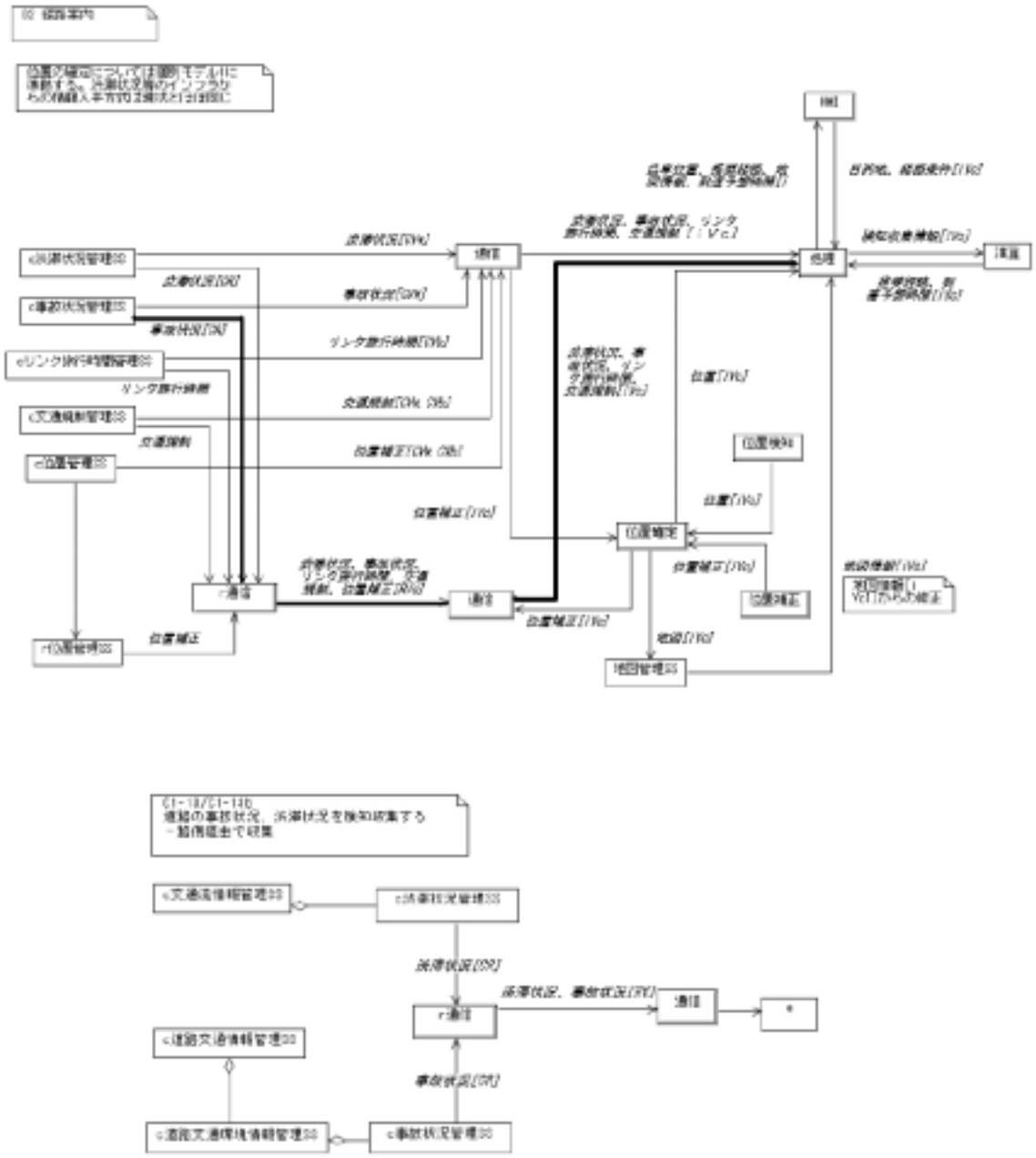


図 3.1: 方式モデルが個別モデル上で変更された例

– 図の下側の方式モデルの c 交通流情報管理 SS、c 道路交通情報管理 SS、c 道路交通環境情報管理 SS が、図の上側の個別モデルでは現れていない。

- 方式モデルの構成要素と対応する個別 MFD の構成要素は一致する。

方式モデルと個別 MFD の間で以上の整合条件を満たさない場合不整合となり、モデル間の一貫性は保たれない。

3.3 不整合の例

以下に方式モデルと個別 MFD の間の不整合の例をあげる。
図 3.2 は方式モデルと個別モデルの間に不整合が現れている例である。図の上段は個別モデル、図の下段は方式モデルである。個別モデルで太線結ばれている個所が方式モデル C1-11c に相当する部分である。方式モデルでは「c 交通規制管理 SS」から「交通規制」という情報が「r 通信」または「通信」ユニットを介し、その後「交通規制管理 SS」を通り「＊」へ渡されることが示されているが、個別モデルでは「交通規制管理 SS」が存在していない。このようにモデル間に対応関係がありながらモデルの要素間に対応関係が結べない場合は不整合となる。

また、次のような場合も不整合となる。図 3.2 では、方式モデルと個別モデルの太線で示した部分を比べてみると、料金所情報のデータフローで運ぶ情報名が方式モデルと制御モデルで食い違っていることがわかる。このような場合、要素は存在しているが名称が異なるために要素間の対応関係を結ぶことができない。よってモデルの一貫性を保てないので不整合となる。

要素間を結ぶデータフローについても不整合が発生する場合がある。図 3.4 はデータフローに不整合が生じている例である。方式モデルでは「地図 [iVc]」というデータフローは「地図管理 SS」から「「位置確定」に向かってデータフローが結ばれているが、個別 MFD では「位置確定」から「地図管理 SS」へ向けてデータフローが結ばれている。つまりデータフローの向きが逆になっている。このような場合もモデルの一貫性が保たれていないので不整合となる。

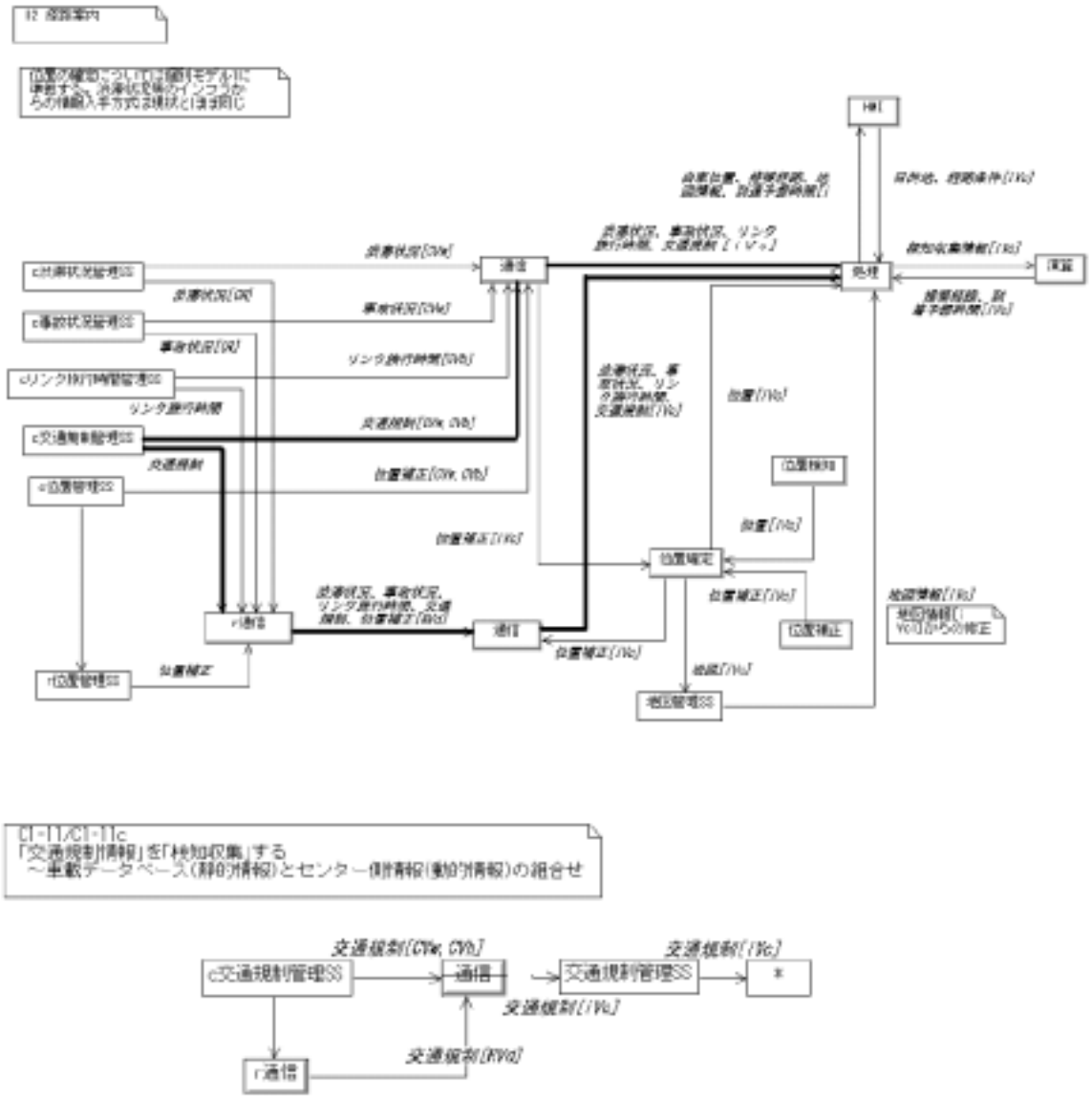


図 3.2: 不整合の例１

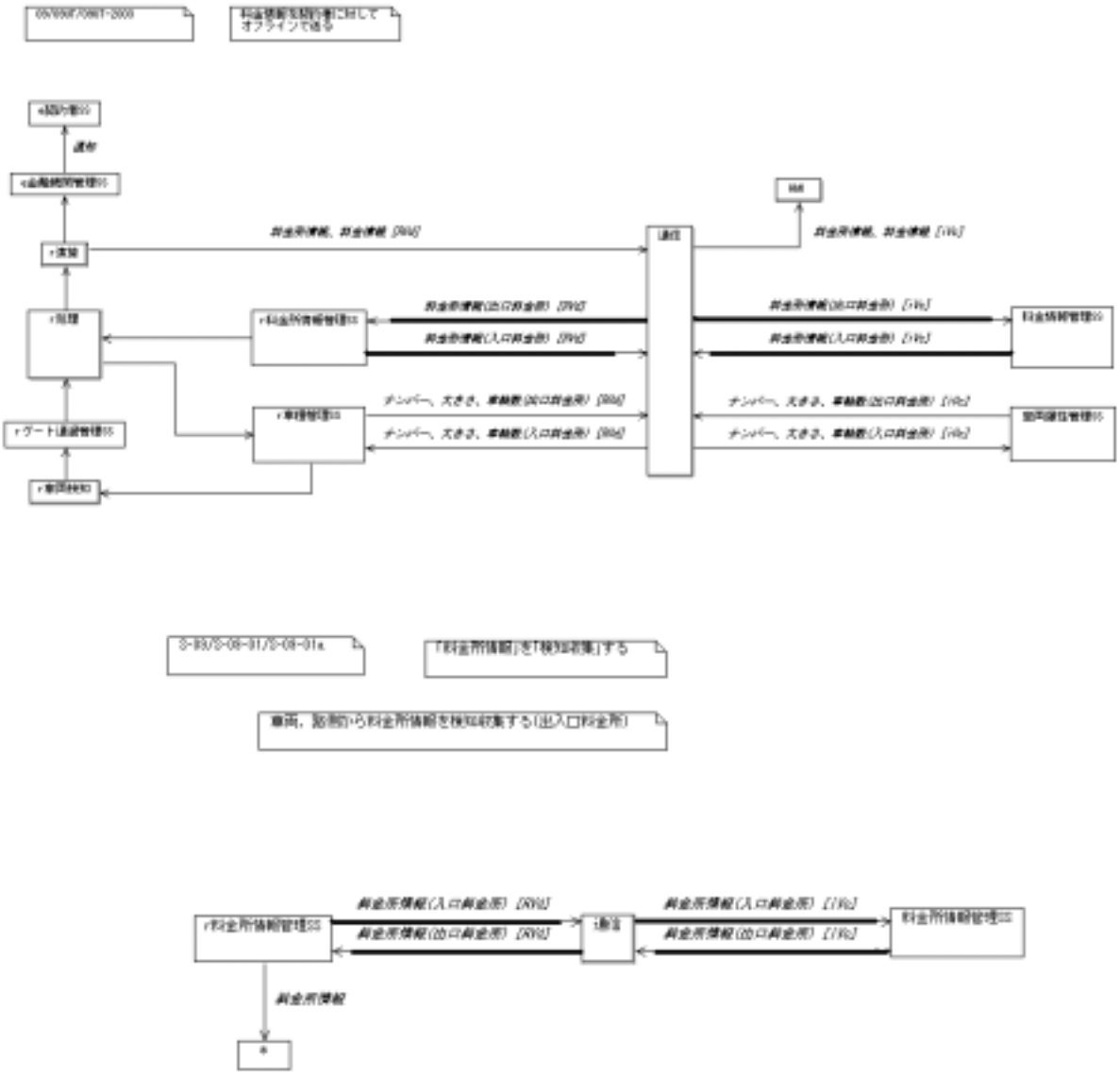


図 3.3: 不整合の例２

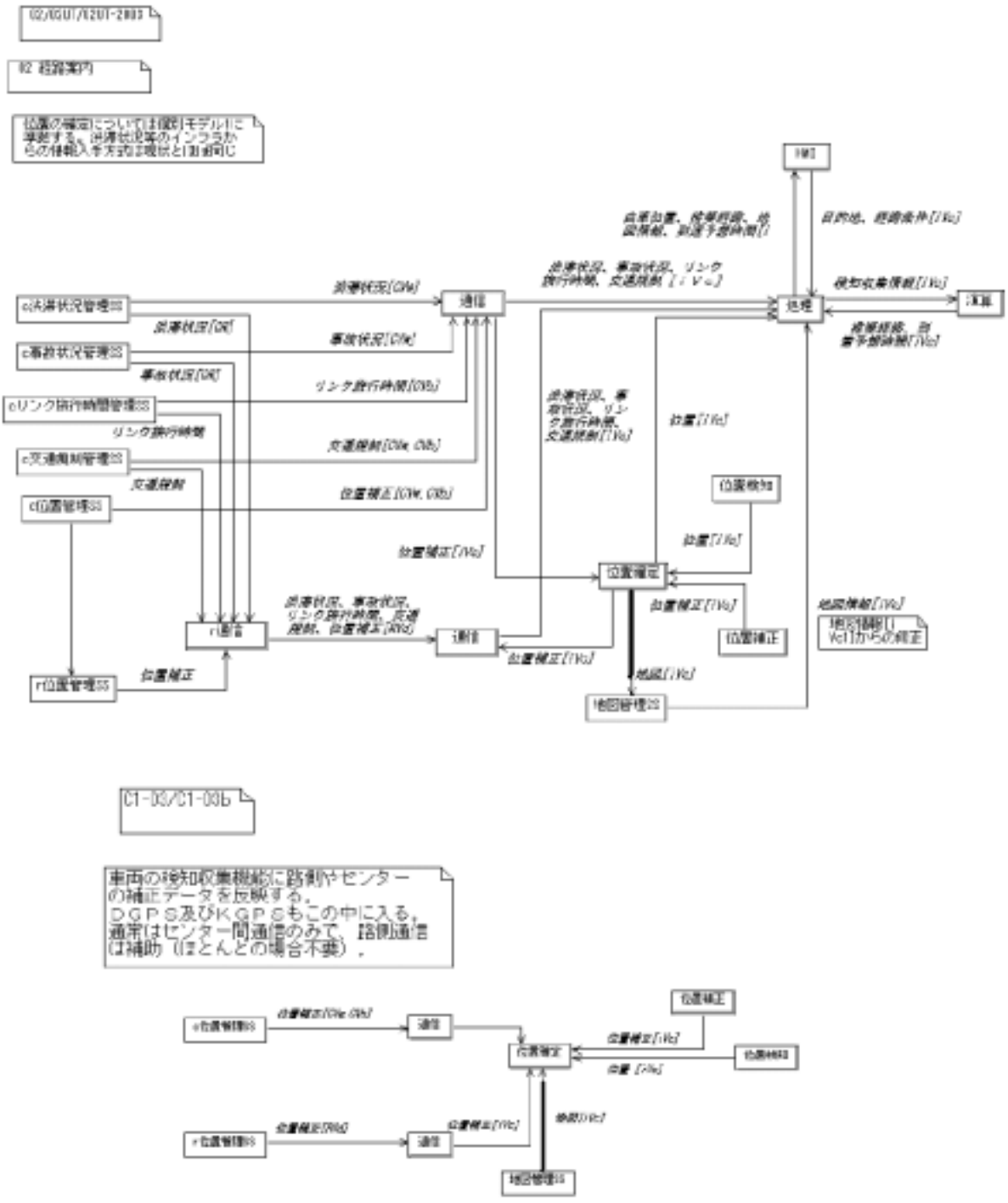

02/IGU/I201-2003
02 経路案内
位置の補正については確認モデルに準拠する。渋滞状況等のインフラからの情報入手方式は現状と1対1同じ
出発位置、検索経路、地図情報、到着予想時間
目的地、経路条件[Vc]
渋滞状況、事故状況、リンク旅行時間、交通規制[Vc]
検知収集情報[Vc]
渋滞状況[CVe]
c渋滞状況管理
通信
処理
渋滞状況[CVc]
事故状況[CVe]
c事故状況管理
事故状況[CVc]
リンク旅行時間[CVb]
cリンク旅行時間管理
リンク旅行時間
交通規制[CVe, CVb]
c交通規制管理
交通規制
位置補正[CVe, CVb]
c位置管理
渋滞状況、事故状況、リンク旅行時間、交通規制[Vc]
位置[Vc]
位置検知
位置[Vc]
位置補正[Vc]
位置補正
位置補正[Vc]
地図情報[Vc]
地図情報はVc1からの補正
渋滞状況、事故状況、リンク旅行時間、交通規制、位置補正[MVc]
r1通信
1通信
位置補正[Vc]
位置補正
r位置管理
地図[Vc]
地図管理
C1-03/C1-03b
車両の検知収集機能に路側やセンターの補正データを反映する。
ＤＧＰＳ及びＫＧＰＳもこの中に入る，通常はセンター間通信のみで，路側通信は補助（ほとんどの場合不要）．
位置補正
位置補正[Vc, CVc]
c位置管理
通信
位置補正[Vc]
位置補正
位置[Vc]
位置検知
位置補正[CVc]
位置補正[Vc]
地図[Vc]
r位置管理
通信
地図管理

図 3.4: 不整合の例３

現在のモデリング手法では方式採用シートで採用された方式モデルは制御モデルで抽出された機能と情報を実現していることが保証されている。方式モデルの各要素の持つ属性や振る舞いについては規定されていない。また、個別 MFD と方式モデルの間では、方式モデルの構造が個別モデルに現れることのみが規定されている、そのため、方式モデルと個別モデルの間でモデル図上で検証できる不整合は本章で例に示したように記述の間違えに限定される。

3.4　異なる抽象度で記述されたモデル間における不整合

現在の記法では、制御モデルはノードとアークの組合わせがどのような役割を持っているかを次に示す５つのパターンで表現するだけにとどまり、その役割を実現するための構造についての制約は記述されていない。

- 検知収集（情報）
  - 入力：情報、もしくは情報を派生するための情報
  - 出力：情報
- 情報提供（情報）
  - 入力：情報
  - 出力：情報（提供対象に適した形態に変換する。）
- 指示制御（指示）
  - 入力：指示を行う判断となる情報（制御モデルに記述）
  - 出力：制御指示（情報ではない）
- 通知警告（警告内容）
  - 入力：警告を行う判断を行う情報（制御モデルに記述）
  - 出力：警告内容（意味対応）
- 確認
  - 入出力の対応なし

また、現状では制御モデルと方式モデルの間は明確な対応付けがなされていないことがモデル作成者によって明らかになったため、モデル間の不整合の検証を行うためにはモデル間の対応を明確にする必要がある。制御モデルと方式モデルの間の対応付けとその対応付けを用いた整合性検証法に関しては 5 章で述べる。

第4章　不整合検証法

4.1　方式モデル-個別MFD間の不整合

物理アーキテクチャを構成するモデルは、実装を考慮しないレベルでの実現方式をモデル化したものである。そのためモデル図上で表現されているのはITSサービスを実現するために必要であると判断された構成要素の物理的な配置だけにとどまる。そのため本研究では方式モデル-個別MFD間における構成要素の物理的な配置にみられる不整合として、モデル間の記述の違いをあげた。本章ではモデル中の各要素間の関係を明確にし、その上で記述上の不整合を検証する方法を提案する。

4.1.1　モデル間の対応関係

方式モデル、個別MFDの各要素の対応は以下のようになっている。

方式モデルと個別MFDの対応関係

- ノードの対応
  - ノードはクラス、サブシステム、ユニットの3種類である。
  - 方式モデルのクラス、ユニット、サブシステムと個別モデルのクラス、ユニット、サブシステムはそれぞれクラス名、ユニット名、サブシステム名によって一意に識別される。
  - 方式モデルと個別モデルの間でノードの名称が同じときそれらのノードは同一のものであると解釈される。
  - 方式モデルのなかの「＊」という名称のノードはそれが任意のノードであることを表わしている。個別モデルのどの名称のノードとも一致する。
  - 個別モデルの「処理」という名称のノードはそれが任意のノードであることを表わしている。方式モデルのどのノードと一致するかは明確に定まっていない。
- アークの対応
  - アークは方向を持ったデータフローである。

- データフローはデータフロー名と両端のノードによって一意に識別される。
- 方式モデルのアークと個別モデルのアークの間では方式モデルのデータフロー名が個別モデルのデータフロー名の一部になっている場合も同一のデータフロー名であると解釈される。
- データフロー名がない場合、そのアークは両端のノード名が一致するアークと同一であると解釈される。
- 方式モデルのアークと個別モデルのアークの間でアークのデータフロー名と両端のノードが同じとき方式モデルのアークと個別モデルのアークは同一のものであると解釈される。

- 方式モデルと個別モデルの関係
  - 個別モデルには方式モデルに含まれているすべてのノードが含まれる。
  - 個別モデルには方式モデルに含まれているすべてのアークが含まれる。

4.1.2 不整合検出の手法

個別モデルには方式選択リストで選択されたすべての方式モデルが含まれていなければならない。よって整合性を調べるためには方式選択リストで選択されたすべての方式モデルと個別モデルの間で、ノードとアークの対応を調べる。

方式モデルと個別 MFD の間で整合がとれていなければならない部分は以下の点である。

- 方式モデルの構造が個別 MFD の構造の 1 部になっている。
  - 方式モデルを構成するユニット、サブシステム、データフローと同一のユニット、サブシステム、データフローが個別 MFD の構成要素含まれている。
  - 方式モデルにおけるユニット、サブシステム、データフローの配置と同一の配置が個別 MFD に含まれている。

よって整合性の検証は方式モデル、個別 MFD の各要素の対応関係にしたがって、両モデル間におけるノードとアークの同一性を調べることで行うことができる。

第5章　抽象度が異なるモデル間の整合を検証するための制約記述法

現在のモデリング手法では制御モデルと方式モデル間のように抽象度の異なる記法で記述されたモデル間の整合についてモデル図上で検証することは大変困難である。これは制御モデルの各機能や情報の実現構造として方式モデルを作成するときの作成過程に大きな要因がある。方式モデルはモデル作成者および、作成に関わった自動車や通信分野などの多くの専門家の知識と経験に基づいて作成されている。しかし、5 年、10 年先に実現されるであろう技術を想定して作成されているため、作成されたモデルが要求仕様を満たしているかどうかを正確に判断するための判断材料は極めて少ない。そのため方式モデルはサブシステムまたはユニットという実相レベルのモデルで使用されるモデルの構成要素比べ非常に抽象度の高い構成要素の物理的な配置を表現することを主な目的としたモデルになっており、ITS 車載システムアーキテクチャにおいて作成された方式モデルが要求仕様を満たすことは前提条件となっている。制御モデルで抽出された特定の機能を方式モデルが満たしているかどうかを検証するための手段として、方式評価シートおよび方式採用リストが存在しているが、これらの資料はあらかじめ規定した評価項目に対する各方式モデルの評価を数字で表現するため評価担当者の主観に大きく左右される。そのため、制御モデルから方式モデルを作成し該当する方式モデルを選択するというモデルの開発過程はモデルの作成と評価に関与した人間の考え方に大きく依存しており人間の主観という曖昧さを多く含む可能性がある。制御モデルから方式モデルを作成する過程の曖昧さを取り除き、モデル間の一貫性のとれた方式モデルを作成するためには、制御モデル、方式モデルそれぞれに現在の MFD 記法より詳細な記述を追加する必要がある。本研究では、現在の MFD 記法に OCL[1][5] を加え、制御モデル、方式モデルへの適用することについて検討した。

5.1　OCL の概要

OCL はモデルに対する制約を記述する言語である。制約は自然言語でも記述されるが、より曖昧性を排除するために IBM 社で開発された言語が OCL である。OCL は、強い数学的なバックグラウンドがなくとも使用できるように設計されている。また、OMG Unified Modeling Language Specification[4] の UML Semantics では、Well-formedness rules は OCl で記述されており、UML において標準の制約記述言語となっている。OCl は IBM 社と

[1] Object Constraint Language

ObjecTime 社によって UML の構成要素として OMG に提案され、UML1.1 から導入された。特徴として、OCL は型付きの言語であること、OCL 式が評価される際に副作用を伴わないこと、プログラミング言語ではないので、プログラミングロジックやフロー制御を記述できないこと等があげられる。OCL の使用目的を以下に示す。

- OCL の使用目的
  - クラス図の不変条件を記述する。
  - ステレオタイプの不変性を記述する。
  - 操作とメソッドの事前、事後条件を記述する。
  - ガード[2]を記述する。
  - ナビゲーション[3]言語として使用する。
  - 操作の制約を記述する。

5.2 制御モデルと方式モデルの対応関係

制御モデルは複数の機能の集まりで構成されている。方式モデルは制御モデルが持つ複数の機能の中のある特定の１つ機能の実現構造を表わしている（図 5.1。そのため、１つの方式モデルは制御モデルの特定の一部と対応している。

制御モデルと方式モデルの間には次のような２つの対応関係が考えられる。

制御モデルと方式モデルの対応１

方式モデルは、制御モデルの１つの機能に対してその実現方式をモデル化したしたものである。そのため、大まかに対応付ける場合は、図 5.1 を見たそのままのように制御モデルの１つの機能に対して１つの方式モデルがそのまま対応つくことになる。この場合、各モデル内の構成要素間に明確な対応はない。例えば、制御モデルの「車両の位置」を表わすノードと方式モデルの中のどのユニットやサブシステムが対応ついているかはわからない。

[2]ガードは動的モデルにおいて、発火の際に満たさなければならない条件のことである。

[3]ナビゲーションとは、要素間に関連が存在するときに一方の要素から関連先の要素のプロパティを参照することである。

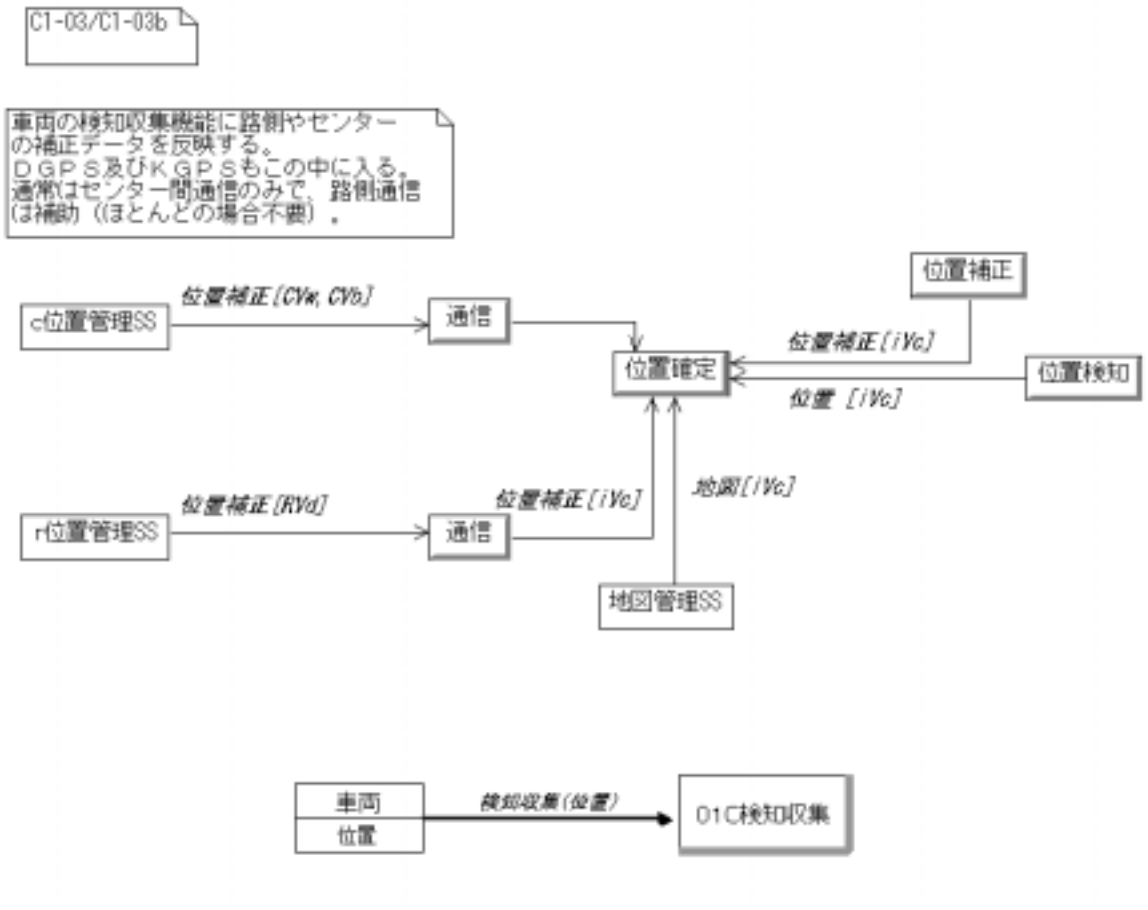


図 5.1: 制御モデルの一部と方式モデルの対応

制御モデルと方式モデルの対応２

制御モデルのなかの、1 対のノードとその間を結ぶデータフローは１つの機能を表わしている。図 5.1 の例では、「車両の位置」を表わすノードと「01c 検知収集」の間を「検知収集（位置）」というデータフローで結ぶことで、「車両の位置情報を検知収集する」という機能を表わしてる。「検知収集」は制御モデルを作成するときに規定された５つのパターンのうちの１つであるので、

- 検知収集（情報）
  - 入力：情報、もしくは情報を派生するための情報
  - 出力：情報

ということを表わしてる。これは図 5.1 の制御モデルを例にすると、「車両の位置」は「位置」という情報、もしくは「位置」という情報を出力するすべてのノードを意味する。そして「01c 検知収集は」は「位置」という情報もしくは「位置」という情報を派生するための情報を入力に持ち、「位置」という情報を出力に持つノードを意味する。この法則に当てはめて図 5.1 の制御モデルの一部と方式モデルの対応を考えると図 5.2 のようになる。この時方式モデルの「通信」ユニットは情報の単なる通り道として解釈されるため対応関係には含まれない。

5.2.1 OCL による制約記述

OCL は UML のクラス図の各クラス毎に制約を記述することができる。その制約は記述された制約にかかわっているクラスのインスタンスに対して適用される。そのため、抽象度が異なる。

制御モデルと方式モデルの間には 2 種類の対応の付け方をすることができたが、「制御モデルの１つの機能に対して１つの方式モデルをそのまま対応付ける」という 1 番目の対応の付け方では、サブシステムやユニットといった個々のノード同士の対応がはっきりしないため、OCL で制約を記述することができない。よって、2 番目の方法で対応をつけた制御モデルと方式モデルの各ノードに対する制約を OCL で記述する。制御モデル、方式モデルをそれぞれパッケージとして解釈することで、オブジェクトの出現条件を OclType を参照する式で書くことができる。また、ナビゲーションを用いてナビゲーション先のオブジェクトの出現条件を書くことで、制御モデルのデータフローと方式モデルの特定のデータフローの間に制約を書くことができる。このような制約を書くことで制御モデルのノードとデータフローと方式モデルのノードとデータフローの整合を検証することが可能になる。

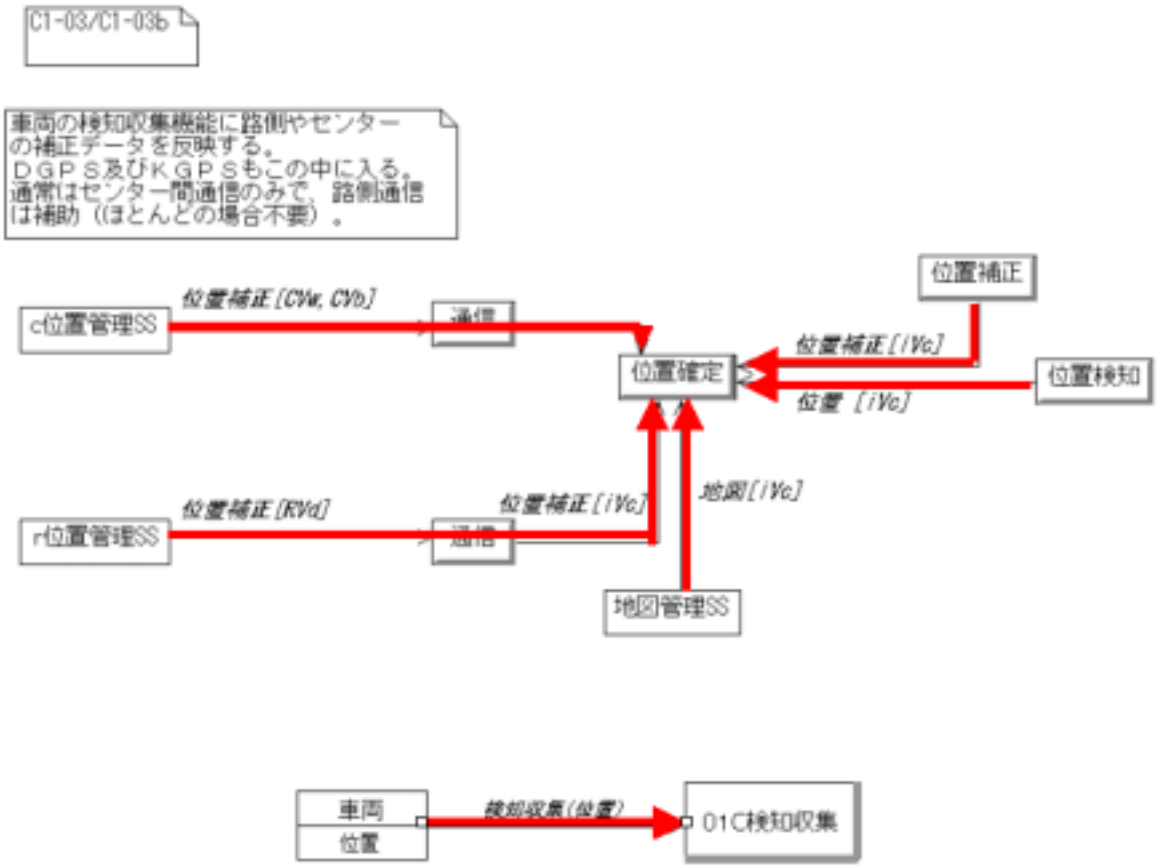


図 5.2: 制御モデルの一部と方式モデルの対応２

- 図の制御モデルはそれぞれ以下の方式モデルのオブジェクトと対応する。
  - 「c 位置管理 SS」と「位置確定」ユニットの間のデータフロー
  - 「r 位置管理 SS」と「位置確定」ユニットの間のデータフロー
  - 「地図管理 SS」と「位置確定」ユニットの間のデータフロー
  - 「位置補正」ユニットと「位置確定」ユニットの間のデータフロー
  - 「位置検知」ユニットと「位置確定」ユニットの間のデータフロー

第6章　計算機支援環境

ITS 車載システムアーキテクチャでは、論理アーキテクチャと物理アーキテクチャの中に複数のモデルが作成されている。それぞれのモデルはモデル単体では十分に検討されているが、モデル間の関係については「専門的な知識を有する者が見れば理解できる」程にしか関係付けられていない。また、各モデルの記述方法が明確に定められておらず、モデル作成者はモデルの本質的な意味を変えない程度に自由にモデルを作成することが許されているため、モデルの記述方法が作成者ごとに異なっている。そのためモデル間に起きている記述の間違いを手作業で探すことが困難になっている。そこで前章で明らかにしたモデル間の対応関係を元に、記述上の不整合を検証するための計算機による支援環境を提案した。

6.1　ツールの概要

ツールは以下の理由から Rational 社の Rose Script と MicroSoft 社の Excel を用いて作成した。

- Rational Rose は UML モデリングツールとして一般的に広く普及している。また、作成したモデルにアクセス可能なスクリプト言語 Rose script の開発環境が付属している。
- Excel は一般的に広く普及している。様々なプロパティ、メソッドの用意された VBA を使用することで容易にマクロを作成することができる。

本研究で実例として用いたモデルは Rational 社の Rose で記述されている。そこで Rose 上のデータを Excel 上にマッピングし、Excel データを検証することで不整合を検出するツールを作成した。このツールでは、はじめに Rational Rose で作成されている方式モデル、個別モデル双方のモデルの構造を、Rose script を用いて Excel データに変換する。つぎに Excel 上のデータをモデル間の対応関係にしたがって比較し、モデル間の不整合個所を特定する。ツールの概要を図 6.1 に示す。

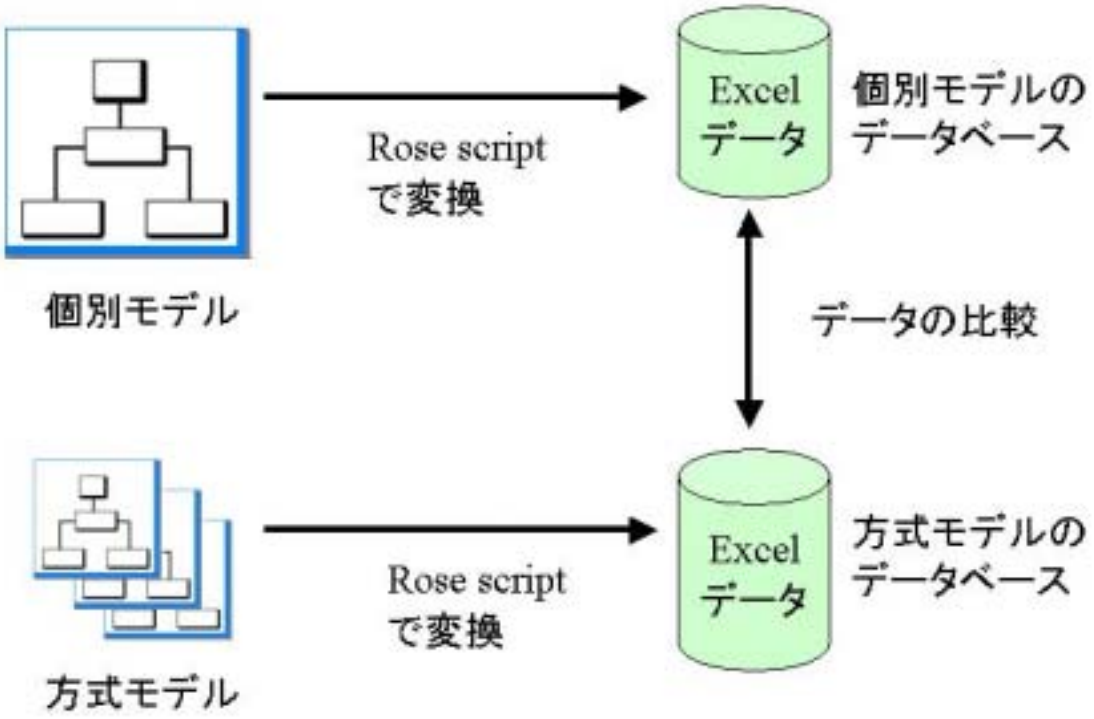


図 6.1: ツールの概要

6.2 Rational Roseから Excelへのデータ変換

Rational Rose を用いて作成された各モデルのモデル構造を Excel データに変換するために、Rose script による変換プログラムを作成した。このプログラムでは Rational Rose でクラス図を作成するときにつくられる各クラスや関連の定義情報の中から、整合検証に必要なデータを抽出し、Excel 上にマッピングした。方式モデル、個別モデルの各モデルからマッピングしたデータを次に示す。

- クラスの定義情報から抽出したデータ
  - モデル名：クラスがどのモデルに含まれているかを示す。
  - クラス名：ユニットおよびサブシステムの名称。クラス名の最後に「SS」と書かれているものはそのクラスがサブシステムであることを示す。それ以外はそのクラスがユニットであることを示す。

- 関連の定義情報から抽出したデータ
  - モデル名：関連がどのモデルに含まれているかを示す。
  - 関連名：データフローの名称
  - Role1 クラス名：データフローにおけるデータの出力先の名称
  - Rose2 クラス名：データフローにおけるデータの出力元の名称

図 6.2 は変換前のクラス図、図 6.3 は変換プログラムを用いて変換した Excel データの例である。

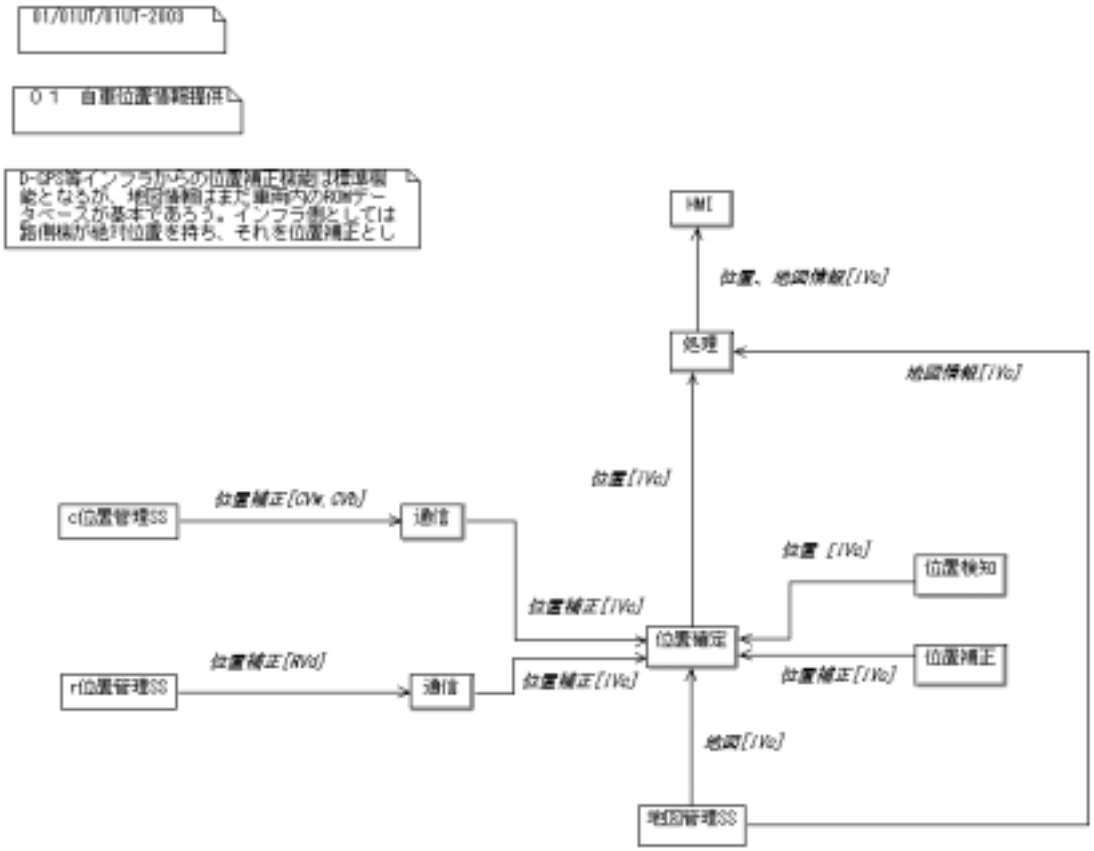
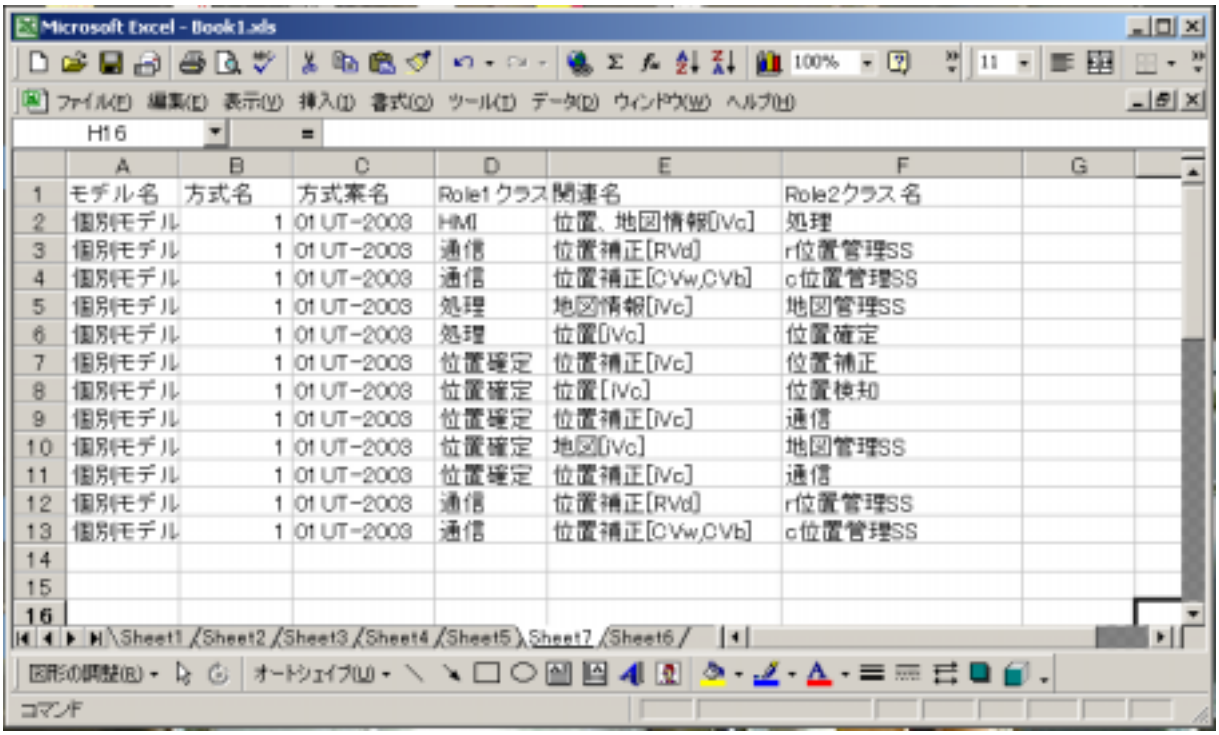


図 6.2: 変換前のクラス図

6.3　データ比較による不整合検証

方式モデルの構造が個別モデルに現れているかどうかを検証するために以下の事柄について比較する。



	A	B	C	D	E	F	G
1	モデル名	方式名	方式案名	Role1 クラス	関連名	Role2クラス名	
2	個別モデル	1	01 UT-2003	HMI	位置、地図情報[Vc]	処理	
3	個別モデル	1	01 UT-2003	通信	位置補正[RVd]	r位置管理SS	
4	個別モデル	1	01 UT-2003	通信	位置補正[CVw,CVb]	c位置管理SS	
5	個別モデル	1	01 UT-2003	処理	地図情報[Vc]	地図管理SS	
6	個別モデル	1	01 UT-2003	処理	位置[Vc]	位置確定	
7	個別モデル	1	01 UT-2003	位置確定	位置補正[Vc]	位置補正	
8	個別モデル	1	01 UT-2003	位置確定	位置[Vc]	位置検知	
9	個別モデル	1	01 UT-2003	位置確定	位置補正[Vc]	通信	
10	個別モデル	1	01 UT-2003	位置確定	地図[Vc]	地図管理SS	
11	個別モデル	1	01 UT-2003	位置確定	位置補正[Vc]	通信	
12	個別モデル	1	01 UT-2003	通信	位置補正[RVd]	r位置管理SS	
13	個別モデル	1	01 UT-2003	通信	位置補正[CVw,CVb]	c位置管理SS	
14							
15							
16							

図 6.3: Excel データの例

- 方式モデルのクラスに対応するクラスが個別モデルに現れているか。
- 方式モデルのデータフローに対応するデータフローが個別モデルに現れているか。
- 方式モデルと個別モデルの間でデータフローの両端のクラスが一致しているか。

ツールでは Excel の１つのブックは個別モデルのデータを収めたシートとその個別モデルで使用される方式モデルのデータを収めたシートで構成されている。そして個別モデルのシートと方式モデルのシートを順に比較し、結果を返す。不整合が見つかった場合はシート上のその場所をカラーで示し、メッセージボックスで不整合があったことを知らせる。実行例を図 6.4 に示す。

6.4 結果と評価

このツールを使用して１０個の個別モデルについて整合性の検証を行った。あらかじめ手作業によって確認されていた不整合はデータの抜けによるものが 8 箇所、データフローに関するものが 2 箇所、データの名称の間違いが 14 箇所であった。ツールで確認できたのはデータの抜けが 8 箇所、データフローが 2 箇所、名称の間違いが 6 箇所であった。名

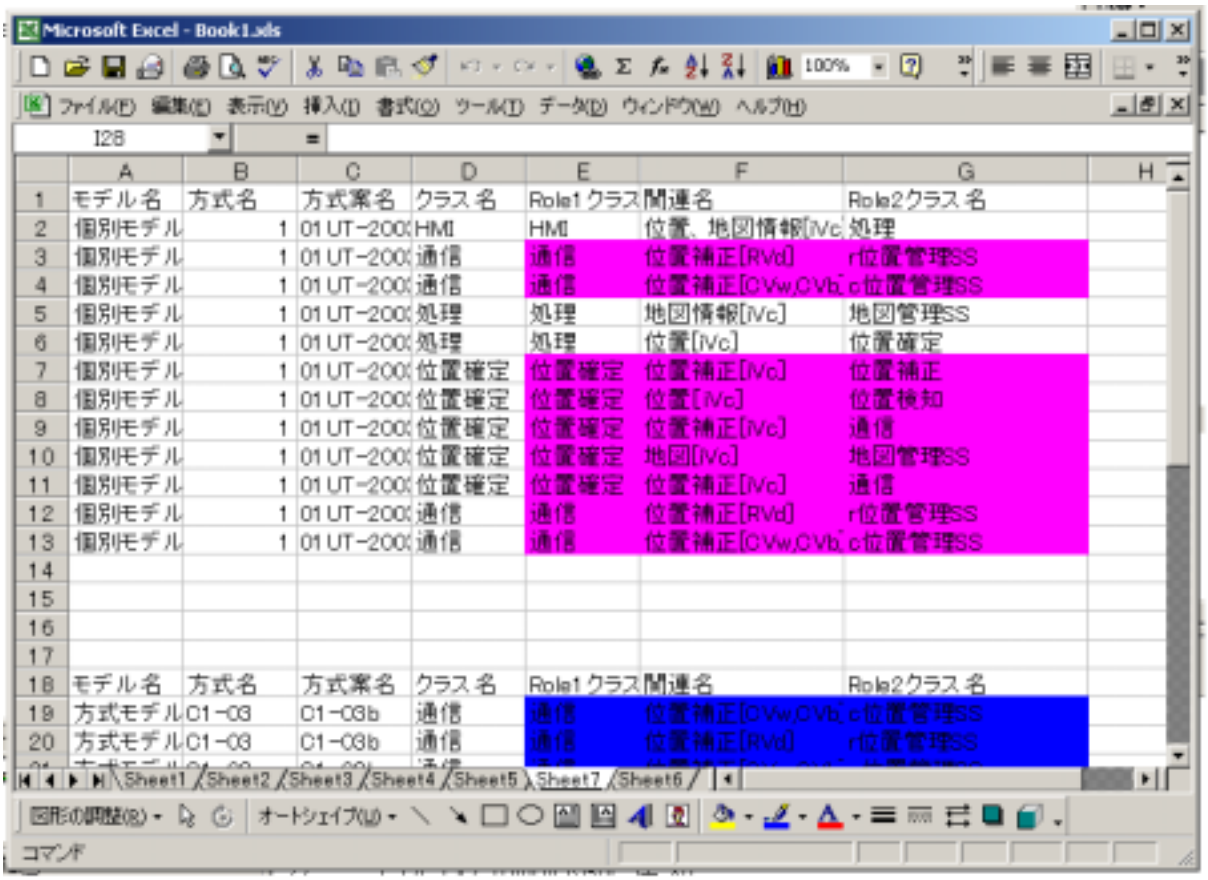


図 6.4: ツールの実行例

称の間違いで検証できなかった 8 箇所はいずれもデータフロー不整合に関するところであった。

第7章　おわりに

7.1　考察

抽象度が同じモデル間における整合性について

抽象度が同じモデル間における整合性を検証するために、方式モデルと個別モデルをれいにあげ、対応関係のあるモデル間における記述上の不整合について検討した。その結果、以下のような不整合が数多く存在することが確認された。

- ユニットやサブシステムといった構成要素の名称の書き間違え。
- ユニットであるべきものがサブシステムとして書かれている。
- 方式モデルで書かれていたユニットやサブシステムが個別モデルに書かれていない。
- データフローの向きが方式モデルと個別モデルでは異なっている。

いずれも非常に単純で、方式モデルと個別モデルを付き合わせて比較すればすぐに間違いであることが確認できるものばかりである。しかし、ITS のような非常に大規模なシステムでは、分析の初期段階でも要求仕様から抽出されるオブジェクトの数も、オブジェクトを用いて作成されるモデルの数も非常に多いため、不整合の起きている場所を特定するのは大変困難である。一般にこのような単純な記述上の不整合を探し、正すという作業は人の手によって行われていることが多いが、多くの工数を要し、修正後の分析モデルにもまだ同様の不整合が残っている可能性がある。事実、本研究で使用した分析モデルはすでに人の手によって修正が加えられたあとのものであったが、多くの不整合が確認された。本研究では単純な対応関係に基づいて支援ツールを作成して不整合の検証を行い、上記の不整合が既存のアプリケーションによるプログラムで簡単に見つけ出せることが確認できたが、それも分析モデル全体の整合性からいえば問題を切り分けただけにすぎない。

抽象度が異なるモデル間における整合性について

抽象度が異なるモデル間における整合性を検証するために、制御モデルと方式モデルを例にあげ、モデル間の不整合について検討した。制御モデルと方式モデルの間の対応関係を明確に書き表す道具として OCL を用いる方法を提案したが、現状では、制御モデルに OCL で制約を記述し、方式モデル上に出現してなければならないオブジェクトを決定す

ることしか出来ない。現在の記法ではユニット、サブシステムは特定の実現技術や実装を想定したものではないので、プロパティやメソッドを持っていない。よって OCL でプロパティやメソッドに関する制約を書くことができないためである。OCL は通常オブジェクトのインスタンスに対する制約を書くのに用いられるので、プロパティの値に関する制約など様々な制約を書くことができる。プロパティやメソッドをもつクラスを用いて作成された分析モデルであれば抽象度の異なるモデル間においてもより詳細な整合についても検証することができる。

7.2 まとめ

オブジェクト指向分析法を大規模システムの分析設計の初期段階に適用し、作成された ITS 車載システムアーキテクチャを実例として取り上げ、作成されたモデル間における整合性について検証した。同じ抽象度で作成されたモデルとして方式モデルと個別モデルを例にあげ、現状で確認できる不整合を検証し、計算機によって支援可能な検証法を提案した。異なる抽象度で作成されたモデルとして制御モデルと方式モデルを例にあげ、不整合を検証するためにモデル間の対応関係を明らかにした。明確にした対応関係を用いて不整合を検証するために、モデル間に OCL による制約を記述し、不整合を検出する方法について検討した。

謝辞

本研究を行なうにあたり終始御指導を賜った片山卓也教授に深く感謝の意を表します。また、権藤克彦助教授、青木利晃助手には大変適切で有用な御助言をいただきました。岡崎光隆氏は日頃のちょっとした質問や相談にも快く応じていただきました。NEC エレクトロンデバイスの岸 知二氏は、ITS の分析モデルを提供していただきました。また、たくさんの質問にも快く応じていただき、大変親切な返答をいただきました。心から感謝致します。
最後に、本論文をまとめるに当たって有意義なご意見と楽しい研究環境を整備していただいたソフトウエア基礎講座の皆様方に深く感謝致します。

2002 年 2 月 15 日 藤本幸伸

関連図書

[1] (財) 自動車走行電子技術協会, ITS システムアーキテクチャの研究「車載物理アーキテクチャの構築」報告書, ITS 規格化,S98-1,1998.

[2] ITS 関係 4 省庁, ITS に係るシステムアーキテクチャ, ITS Japan,1999. http://www.iijnet.or.jp/vertis/SA/J-SA

[3] the Department of Transportation (DOT) , National ITS Architecture Version 3.0, the Department of Transportation (DOT) ,2001. http://www.iteris.com/itsarch

[4] Object Management Group, UML 仕様書, 株式会社アスキー,2001.

[5] Jos Warmer,Anneke Kleppe The Object Constraint Language Addison Wesley Longman,inc,1999.